In the Claims

Amend the claims as follows:

1. (currently amended) A method of parallel processing in a memory structure comprising:

creating a first thread in the memory structure which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process;

providing a second thread in the memory structure which represents an independent flow of control managed by a program structure separate from the first thread;

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread and marking the work as not complete;

if the first thread is in the second state awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is in the first state processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue; and

using the program structure to destroy the first thread in the memory structure after the first thread completes a desired amount of work.

2. (original) The method of claim 1 wherein the second thread continues to place additional work in the queue, and the first thread sequentially processes the additional work in the queue as it completes processing prior work.

3. (cancelled)

4. (original) The method of claim 1 wherein if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

5. (previously presented) The method of claim 1 wherein the first thread is reused to process other work before being destroyed.

6. (cancelled)

7. (currently amended) A method of parallel processing in a memory structure comprising:

creating a first thread in the memory structure which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process;

providing a second thread in the memory structure which represents an independent flow of control managed by a program structure separate from the first thread;

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread, the work placed in the ~~first thread~~ queue being marked as not complete;

if the first thread is in the second state awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is in the first state processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue;

using the second thread to place additional work in the queue;

using the first thread to sequentially process the additional work in the queue as it completes processing prior work; and

using the program structure to destroy the first thread in the memory structure after the first thread completes a desired amount of work.

8. (original) The method of claim 7 wherein if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

9. (previously presented) The method of claim 7 wherein the first thread is reused to process other work before being destroyed.

10. (cancelled)

11. (currently amended) A program storage device readable by a machine, tangibly embodying a program of instructions executable by the machine to perform method steps of parallel processing in a memory structure using i) a first thread which represents an independent flow of control managed by a program structure, said first thread having two states, a first state processing work for the program structure and a second state undispatched awaiting work to process, and ii) a second thread which represents an independent flow of control managed by a program structure separate from the first thread, said method steps comprising:

creating the first thread;

using the second thread to prepare work for the first thread to process;

placing the work prepared by the second thread in a queue for processing by the first thread and marking the work as not complete;

if the first thread is in the second state awaiting work to process when the work prepared by the second thread is placed in the queue, dispatching the first thread and using it to process the work in the queue;

if the first thread is in the first state processing other work when the work prepared by the second thread is placed in the queue, using the first thread to complete processing of the other work, access the work in the queue, and then process the work in the queue; and

using the program structure to destroy the first thread in the memory structure after the first thread completes a desired amount of work.

12. (original) The program storage device of claim 11 wherein, in the method steps, the second thread continues to place additional work in the queue, and the first thread sequentially processes the additional work in the queue as it completes processing prior work.

13. (cancelled)

14. (original) The program storage device of claim 11 wherein, in the method steps, if the first thread is processing other work when the work prepared by the second thread is placed in the queue, and when the first thread completes processing of the work in the

queue, using the first thread to mark the completed work as complete, wherein subsequent work from the second thread is made to wait until the previous work in the first thread is marked complete.

15. (previously presented) The program storage device of claim 11 wherein, in the method steps, the first thread is reused to process other work before being destroyed.

16. (cancelled)

17. (previously presented) The method of claim 1 including providing a memory structure having a plurality of planes, each plane having a context comprising: a) a thread representing an independent flow of control managed by a program structure, b) a heap portion for data structure, and c) a plurality of stack portions for function arguments, each thread using a different stack portion, and wherein only one of the first and second threads uses a context at any particular time.

18. (currently amended) The method of ~~claim 1~~ claim 7 including providing a memory structure having a plurality of planes, each plane having a context comprising: a) a thread representing an independent flow of control managed by a program structure, b) a heap portion for data structure, and c) a plurality of stack portions for function arguments, each thread using a different stack portion, and wherein only one of the first and second threads uses a context at any particular time.

19. (previously presented) The program storage device of claim 11 wherein the memory structure includes a plurality of planes, each plane having a context comprising: a) a thread representing an independent flow of control managed by a program structure, b) a heap portion for data structure, and c) a plurality of stack portions for function arguments, each thread using a different stack portion, and wherein only one of the first and second threads uses a context at any particular time.